

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	37	("5432795" "5528753" "5581696" "5613063" "5664191" "5732273" "5734908" "5758154" "5790858" "5958010" "5987249" "6186677" "6314558").PN. OR ("6760903").URPN.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/11/27 15:15
S1	848	717/124.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/11/27 15:15
S2	589	717/127.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/11/21 10:44
S3	268	717/129.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/11/21 10:44
S4	387	717/130.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/11/21 10:44
S5	25	breakpoint and (prob\$3 or instrument\$5 or hook\$3 or debug\$4) near5 (bytecode or java or "virtual machine" or vm or jvm) and (memory or heap or stack or shared or global) and (prob\$3 or instrument\$5 or hook\$3 or debug\$4) near5 source and (pars\$3 or ast or intermediate or tree)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/11/21 10:52
S6	9	"6961923" "6959441" "6769117" "7047520" "6961923" "6959441" "6769117" ("7047520").pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/11/21 10:53
S7	34	S5 or S6	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/11/21 10:53


[Subscribe \(Full Service\)](#) [Register \(Limited Service\)](#)

Search: ☒ The ACM Digital Library ☐ The USPTO

THE ACM DIGITAL LIBRARY
[Feedback](#) [Report a problem](#)

Published since January 1990 and Published before June 2003

Terms used **interpreter** **breakpoint** **ast**

Sort results by ☒ [Save results to a Binder](#) [Try an Advanced Search](#)
☒ [Search Tips](#) [Try this search](#)
 Display results ☐ Open results in a new window

Results 1 - 3 of 3

1 [PCCTS reference manual: version 1.00](#)

T. J. Parr, H. G. Dietz, W. E. Cohen
 February 1992 **ACM SIGPLAN Notices**, Volume 27 Issue 2
Publisher: ACM Press

Full text available: [pdf\(3.77 MB\)](#)

Additional Information: [full citation](#), [citing articles](#)

2 [Understanding memory allocation of scheme programs](#)

Manuel Serrano, Hans-J. Boehm
 September 2000 **ACM SIGPLAN Notices , Proceedings of the fifth ACM international conference on Functional programming**
 35 Issue 9

Publisher: ACM Press

Full text available: [pdf\(821.49 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

Memory is the performance bottleneck of modern architectures. Keeping memory consumption as low as possible enables fast and unobtrusive applications. We describe techniques to estimate the memory use of programs implemented in functional languages, the complex translations of some high level constructs, and the use of au


managers. To help understand memory allocation behavior of Scheme programs, we designed two complementary tools. The first one reports on frequency of

3 Automating the re-declaration of unneeded globals as private

◆ Amitava Datta, Prabhaker Mateti

March 1993 **Proceedings of the 1993 ACM/SIGAPP symposium on Applications, states of the art and practice**

Publisher: ACM Press




Full text available:  [pdf\(787.97 KB\)](#) Additional Information: [full citation](#), [reference review](#)

Keywords: formal methods, functional languages, software re-engineering, transformations

Results 1 - 3 of 3

The ACM Portal is published by the Association for Computing Machinery
ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)



USPTO

[Subscribe \(Full Service\)](#) [Register \(Limited Service\)](#)

Search: ☒ The ACM Digital Library ☐ The USPTO
☒ +interpreter +compile +ast

THE ACM DIGITAL LIBRARY

[Feedback](#) [Report a problem](#)

Published since January 1990 and Published before June 2003

Terms used **interpreter compile ast**

Sort results by

☒ [Save results to a Binder](#)
[Try an Advanced Search](#)
☒ [Search Tips](#)
[Try this search](#)

Display results

☐ Open results in a new window

Results 1 - 20 of 68

Result page: [1](#) [2](#) [3](#) [4](#) [next](#)

Results

1 [Building incremental programs using partial evaluation](#)

R. S. Sundaresh

May 1991 **ACM SIGPLAN Notices , Proceedings of the 1991 ACM SIGPLAN Conference on Partial evaluation and semantics-based program manipulation**
 Volume 26 Issue 9

Publisher: ACM Press

Full text available: [pdf\(782.47 KB\)](#) Additional Information: [full citation](#), [references](#), [index terms](#)

2 [MaJIC: compiling MATLAB for speed and responsiveness](#)

George Almási, David Padua

May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN Conference on Programming language design and implementation PLDI 2002**
 Issue 5

Publisher: ACM Press

Full text available: [pdf\(619.32 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

This paper presents and evaluates techniques to improve the execution performance of MATLAB. Previous efforts concentrated on source to source translation


compilation; **MaJIC** provides an interactive frontend that looks like MA compiles/optimizes code behind the scenes in real time, employing a cor time and speculative ahead-of-time compilation. Performance results shc mixture of these two techniques can yield near-zero response time as ...

3 Programmable syntax macros

◆ Daniel Weise, Roger Crew

June 1993 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLA on Programming language design and implementation PLD**
Issue 6

Publisher: ACM Press

Full text available:  [pdf\(1.09 MB\)](#) Additional Information: [full citation](#), [abst citings](#), [index ter](#)


Lisp has shown that a programmable syntax macro system acts as an adj that gives the programmer, important and powerful abstraction facilities i language. Unlike simple token substitution macros, such as are provided preprocessor), syntax macros operate on Abstract Syntax Trees (ASTs). syntax macro systems have not yet been developed for syntactically rich C because rich concrete syntax requires the manual con ...

4 Using types to analyze and optimize object-oriented programs

◆ Amer Diwan, Kathryn S. McKinley, J. Eliot B. Moss

January 2001 **ACM Transactions on Programming Languages and Syst**
Volume 23 Issue 1

Publisher: ACM Press

Full text available:  [pdf\(414.51 KB\)](#) Additional Information: [full citation](#), [abst citings](#), [index ter](#)

Object-oriented programming languages provide many software enginee these often come at a performance cost. Object-oriented programs make method invocations and pointer dereferences, both of which are potentia machines. We show how to use types to produce effective, yet simple, te reduce the costs of these features in Modula-3, a statically typed, object- Our compiler performs type-based alias analysis to ...

Keywords: alias analysis, classes and objects, method invocation, object polymorphism, redundancy elimination

5 Array morphology



Robert Bernecky

September 1993 **ACM SIGAPL APL Quote Quad , Proceedings of the i
conference on APL APL '93**, Volume 24 Issue 1

Publisher: ACM Press

Full text available: [pdf\(1.02 MB\)](#) Additional Information: [full citation](#), [abst
citing](#), [index ter](#)

Array morphology is the study of the form, structure, and evolution of an *annotation* for a program written in an applicative array language is an *annotation* for the program, amended with information about the arrays created by the *notations* are useful in the production of efficient compiled code for *appl* programs. Array morphology is shown to be an effective compiler writer *an array annotator in action* are pre ...

6 Technical correspondence: Using smgn for rapid prototyping of small domain-specific languages



Holger M. Kienle

September 2001 **ACM SIGPLAN Notices**, Volume 36 Issue 9

Publisher: ACM Press

Full text available: [pdf\(975.44 KB\)](#) Additional Information: [full citation](#), [abst
index terms](#)

This paper presents *smgn*, a grammar-based tool that provides support for *and* automatic parse tree construction. The parse tree can be easily navigated and manipulated with a specific macro language while conveniently generating *smgn* is easy to learn and well suited for rapid prototyping of small domain-specific languages. It is part of the SUIF compiler system, where it has been used in the development of the *Hoof* domain-specific language. Further ...

Keywords: SUIF compiler system, domain-specific language, domain-specific language prototyping, prototyping


7 Re-targetability in software tools



Premkumar T. Devanbu


September 1999 **ACM SIGAPP Applied Computing Review**, Volume 7

Publisher: ACM Press


Full text available:  [pdf\(756.28 KB\)](#) Additional Information: [full citation](#), [abst](#)

Software tool construction is a risky business, with uncertain rewards. Much is used. This is a truism: software tools, however brilliantly conceived, well meticulously constructed, have little impact unless they are actually adopted by programmers. While there are no sure-fire ways of ensuring that a tool will be used, experience indicates that *retargetability* is an important enabler for wide adoption. In this paper, we elaborate on the need for retargetable compilers.

8 Understanding memory allocation of scheme programs


 Manuel Serrano, Hans-J. Boehm
September 2000 **ACM SIGPLAN Notices , Proceedings of the fifth ACM international conference on Functional programming** |
35 Issue 9

Publisher: ACM Press


Full text available:  [pdf\(821.49 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [index terms](#)

Memory is the performance bottleneck of modern architectures. Keeping memory consumption as low as possible enables fast and unobtrusive applications. This paper describes techniques to estimate the memory use of programs implemented in functional languages, the complex translations of some high level constructs, and the use of automatic memory managers. To help understand memory allocation behavior of Scheme programs, we designed two complementary tools. The first one reports on frequency of memory allocation and deallocation.

9 Clarity MCode: a retargetable intermediate representation for compilation

 Brian T. Lewis, L. Peter Deutsch, Theodore C. Goldstein
March 1995 **ACM SIGPLAN Notices , Papers from the 1995 ACM SIGPLAN conference on Intermediate representations**, Volume 30 Issue 3

Publisher: ACM Press


Full text available:  [pdf\(948.64 KB\)](#) Additional Information: [full citation](#), [citations](#)

10 Maya: multiple-dispatch syntax extension in Java

Jason Baker, Wilson C. Hsieh

- ◆ May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN on Programming language design and implementation PLDI**
Issue 5

Publisher: ACM Press

Full text available:  [pdf\(152.75 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

We have designed and implemented Maya, a version of Java that allows extend and reinterpret its syntax. Maya generalizes macro systems by treating productions as generic functions, and semantic actions on productions as the corresponding generic functions. Programmers can write new generic grammar productions) and new multimethods (i.e., semantic actions), they can extend the grammar of the language and change the semantics of ...


Keywords: Java, generative programming, macros, metaprogramming

11 An extensible programming environment for Modula-3

◆ Mick Jordan

October 1990 **ACM SIGSOFT Software Engineering Notes , Proceedings of the ACM SIGSOFT symposium on Software development environments**
4, Volume 15 Issue 6

Publisher: ACM Press

Full text available:  [pdf\(1.33 MB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)


This paper describes the design and implementation of a practical programming environment for the Modula-3 programming language. The environment provides an extensible intermediate representation of programs and makes extensible components. The environment is implemented in Modula-3 and exploits features of the language.

12 The undergraduate capstone software design experience

◆ Jean R. S. Blair, Eugene K. Ressler, Thomas D. Wagner


November 1997 **Proceedings of the conference on TRI-Ada '97**

Publisher: ACM Press

Full text available:  [pdf\(1.02 MB\)](#) Additional Information: [full citation](#), [references](#), [index terms](#)

13 Technical correspondence: Good design principles in a compiler university


◆ César F. Acebal, Raúl Izquierdo Castanedo, Juan M. Cueva Lovelle

April 2002 **ACM SIGPLAN Notices**, Volume 37 Issue 4**Publisher:** ACM PressFull text available:  [pdf\(604.45 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#)

This paper presents what aims to be an example of good design principle compiler construction. To be more specific, it presents an interpreter of a small, object-oriented language, called SmallScript, that has been designed to be taught in a University course. Our aim is not to develop a new, revolutionary language, but a spectacular advance in some research field of compiler construction. In the future, we aim to offer both students and teachers ...

Keywords: JJTree, JavaCC, compiler design, design patterns, interpreter**14 Mutation analysis using mutant schemata**

◆ Roland H. Untch, A. Jefferson Offutt, Mary Jean Harrold


July 1993 **ACM SIGSOFT Software Engineering Notes**, **Proceedings of the SIGSOFT international symposium on Software testing and analysis '93**, Volume 18 Issue 3**Publisher:** ACM PressFull text available:  [pdf\(872.48 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

Mutation analysis is a powerful technique for assessing and improving the reliability of data used to unit test software. Unfortunately, current automated mutation testing tools suffer from severe performance problems. This paper presents a new method for mutation analysis that uses program schemata to encode all mutants for a given program, which is subsequently compiled and run at speeds substantially faster than achieved by ...

Keywords: fault-based testing, mutation analysis, program schemata, software testing

15 Towards automatic construction of staged compilers


◆ Matthai Philipose, Craig Chambers, Susan J. Eggers

January 2002 **ACM SIGPLAN Notices , Proceedings of the 29th ACM SIGACT symposium on Principles of programming language**
Volume 37 Issue 1**Publisher:** ACM PressFull text available:  pdf(269.51 KB) Additional Information: [full citation](#), [abstracts](#), [citations](#)

Some compilation systems, such as offline partial evaluators and selective compilation systems, support staged optimizations. A staged optimization logically single optimization is broken up into stages, with the early stage preplanning set-up work, given any available partial knowledge about the code to be compiled, and the final stage completing the optimization. The final stage is faster than the original optimization by having much of its work ...

16 Compiling scheme to JVM bytecode:: a performance study

◆ Bernard Paul Serpette, Manuel Serrano

September 2002 **ACM SIGPLAN Notices , Proceedings of the seventh ACM international conference on Functional programming**
Volume 37 Issue 9**Publisher:** ACM PressFull text available:  pdf(298.96 KB) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)


We have added a Java virtual machine (henceforth JVM) bytecode generator and an optimizing Scheme-to-C compiler Bigloo. We named this new compiler Bigloo. We have used this new compiler to evaluate how suitable the JVM bytecode generator is for compiling strict functional languages such as Scheme. In this paper, we discuss the performance issue. We have measured the execution time of many Scheme programs compiled to C and when compiled to JVM. We found that for each benchmark

Keywords: Java virtual machine, compilation, functional languages, scheme**17** Caching function calls using precise dependencies

◆ Allan Heydon, Roy Levin, Yuan Yu

May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN conference on Programming language design and implementation PLDI**

Issue 5


Publisher: ACM PressFull text available:  [pdf\(243.57 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [index terms](#)

This paper describes the implementation of a purely functional program building software systems. In this language, external tools like compilers invoked by function calls. Because some function calls are extremely expensive, it is obviously important to reuse the results of previous function calls whenever possible. Caching a function call requires the language interpreter to record all values that a function call depends. For optimal caching, it is important to know the dependencies of a function call.

18 [Technical correspondence: Implementing a real computational-environment to develop a runtime-adaptable reflective platform](#)



Francisco Ortín, Juan Manuel Cueva

August 2002 **ACM SIGPLAN Notices**, Volume 37 Issue 8**Publisher:** ACM PressFull text available:  [pdf\(714.89 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [index terms](#)


Different techniques are emerging in order to build adaptable computing software engineering methods. Two examples in the software engineering oriented programming and multi-dimensional separation of concerns. The first example is the separation of functional code from reusable crosscutting aspects, creating the final application by weaving the program and its specific aspects. They lack runtime adaptability, simultaneous adaptation. Dynamic adaptability is offered by ...

Keywords: aspect-oriented programming, generic interpreter, meta-object reflection, runtime adaptability

19 [Accurate static estimators for program optimization](#)



Tim A. Wagner, Vance Maverick, Susan L. Graham, Michael A. Harrison
June 1994 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN Workshop on Programming language design and implementation PLDI**
Issue 6

Publisher: ACM PressFull text available:  [pdf\(1.04 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [index terms](#)

MB)citings, index ter


Determining the relative execution frequency of program regions is essential optimization techniques, including register allocation, function instruction scheduling. Estimates derived from profiling with sample input are regarded as the most accurate source of this information; static (compiler) estimates are considered to be distinctly inferior. If static estimates were shown to be correct, however, their convenience would outweigh minor ...

20 Implementing incremental code migration with XML

◆ Wolfgang Emmerich, Cecilia Mascolo, Anthony Finkelstein

June 2000 **Proceedings of the 22nd international conference on Software Engineering**

Publisher: ACM Press

Full text available:  [pdf\(124.85 KB\)](#) Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

We demonstrate how XML and related technologies can be used for code granularity, thus overcoming the restrictions of existing approaches. By using a particular granularity for mobile code, we enable complete programs as well as lines of code to be sent across the network. We define the concept of incremental mobility as the ability to migrate and add, remove, or replace code fragments (increments) in a remote program. The combination of fine-grain ...




Keywords: XML technologies, incremental code migration

Results 1 - 20 of 68

Result page: [1](#) [2](#) [3](#) [4](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery
ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)

[Home](#) | [Login](#) | [Logout](#)
IEEE Xplore
RELEASE 2.1

Welcome United States Patent and
Trademark Office

Search Results

[BROWSE SEARCH](#) [IEEE GUIDE](#)

Results for "(((ast compiler interpreter breakpoint)<in>metadata)) <an
1990 <and> ..."
Your search matched 0 documents.
A maximum of 100 results are displayed, 25 to a page, sorted by **Relevance**
Descending order.

» Search Options

[View Session History](#)
[New Search](#)

Modify Search

☐ Check to search only within this results set

» Key

IEEE JNL

IEEE
Journal or
Magazine

IEEE JNL

IEEE Journal
or Magazine

IEEE CNF

IEEE
Conference
Proceeding

IEEE CNF

IEEE
Conference
Proceeding

IEEE STD

IEEE
Standard

Display Format: ☒ Citation ☐ Citation & Abstract

No results were found.

Please edit your search criteria and try again. Refer assistance revising your search.

Indexed by

 Inspec

[Home](#) | [Login](#) | [Logout](#)


Welcome United States Patent and Trademark Office

Search Results

[BROWSE SEARCH](#) [IEEE GUIDE](#)

Results for "(((ast compiler breakpoint)<in>metadata)) <and> (pyr >= pyr <= 20...)

Your search matched 0 documents.

A maximum of 100 results are displayed, 25 to a page, sorted by Relevance Descending order.

» Search Options

[View Session](#)
[History](#)
[New Search](#)

Modify Search

(((ast compiler breakpoint)<in>metadata)) <and> (pyr >= pyr <= 20...)

☐ Check to search only within this results set

» Key

IEEE JNL

IEEE

Journal or Magazine

IEEE JNL

IEEE Journal or Magazine

IEEE CNF

IEEE

Conference Proceeding

IEEE CNF

IEEE

Conference Proceeding

IEEE STD

IEEE

Standard

Display Format: ☒ Citation ☐ Citation & Abstract

No results were found.

Please edit your search criteria and try again. Refer assistance revising your search.

Indexed by

Inspec

[Home](#) | [Login](#) | [Logout](#)

Welcome United States Patent and Trademark Office

Search Results

[BROWSE SEARCH](#) [IEEE GUIDE](#)

Results for "(((ast probe breakpoint)<in>metadata)) <and> (pyr >= 199 <= 2003))"

Your search matched 0 documents.

A maximum of 100 results are displayed, 25 to a page, sorted by Relevance Descending order.

» Search Options

[View Session](#)
[History](#)
[New Search](#)

Modify Search

(((ast probe breakpoint)<in>metadata)) <and> (pyr >= 199 <= 2003))

☐ Check to search only within this results set

» Key

IEEE JNL

IEEE

Journal or
Magazine

IEE JNL

IEE Journal
or Magazine

IEEE CNF

IEEE

Conference
Proceeding

IEE CNF

IEE

Conference
Proceeding

IEEE STD

IEEE

Standard

Display Format: ☒ Citation ☐ Citation & Abstract

No results were found.

Please edit your search criteria and try again. Refer assistance revising your search.

Indexed by

Inspect